TAREA PARA DWEC04 https://github.com/JuanjoMayorga/Mayorga_Usero_JuanJose_DWEC04.git

UT04 Práctica 1: Objetos listas

1. Listas

Dada la práctica de la UT03.1 para la implementación de Listas y ListasOrdenadas, rediseña el código para que estén implementadas mediante objetos para facilitar su mantenimiento.

Deberás implementar un conjunto de excepciones en forma de objeto para comunicar errores. Los objetos para representar las excepciones deben ser específicos de la práctica, no errores genéricos.

Para los objetos de ListasOrdenadas deben heredar de Lista los métodos que sean iguales, y sobrescribir los que personalicen la funcionalidad.

Haz uso de los recursos que consideres para implementar la práctica, propiedades públicas, campos privados, métodos públicos, si necesitas getter o setter, propiedades o métodos estáticos. No será necesario utilizarlos todos, ni habrá un mínimo de ellos que sean necesario, tan solo que se utilicen con coherencia.

Como en la tarea DWEC03, tenemos tres tipos de listas: una lista, una lista ordenada por ISBN y una lista ordenada por fecha.

Por ello, para este ejercicio he hecho tres "bloques" análogos compuestos por un archivo .html, un archivo .js en el que implemento el modelo de lista en cuestión, y otro archivo .js de testeo en el que efectúo algunas operaciones sobre las listas.

Así, tendré los archivos:

- *Lista.html*
- *Lista.js*
- *Lista-test.js*
- *ListaOrdenadaPorISBN.html*
- *ListaOrdenadaPorISBN.js*
- *ListaOrdenadaPorISBN-test.js*
- *ListaOrdenadaPorFecha.html*
- *ListaOrdenadaPorFecha.js*
- *ListaOrdenadaPorFecha-test.js*

Tendré además unos de archivos comunes a los tres bloques de ejercicios: un archivo .css con una hoja de estilo aplicada sobre los .html, un archivo .js en el que defino varias excepciones en las que se puede incurrir durante la ejecución del código, y un archivo .js en el que modelo los objetos Book, que en la práctica DWEC03 declaré de forma literal en las funciones de testeo.

- *Book.js*
- *Excepciones.js*
- *Mayorga_Usero_JuanJose_DWEC04_hojadeestilo.css*

Muestro el código de cada archivo:

Book.js

```
/*
Juan José Mayorga Usero
CFGS Desarrollo de Aplicaciones Web
I.E.S. Maestre de Calatrava - Ciudad Real
Desarrollo Web en Entorno Cliente
UT04 Práctica 1
*/

/*
He definido un objeto Book con un constructor para crear los libros que se van a almacenar
en Lista.
 */
function Book(isbn, title, author, publicationDate, price)
{
    /* En un primer momento, pense en poner las propiedades públicas. Luego decidí que era
mejor usar campos privados. */
    /*this.isbn = isbn;
    this.title = title;
    this.author = author;
    this.publicationDate = publicationDate;
    this.price = price;*/

    /* Aquí defino los campos privados. */
    let _isbn = isbn;
    let _title = title;
    let _author = author;
    let _publicationDate = publicationDate;
    let _price = price;

    /* Creo métodos get y set para cada uno de los atributos.*/
    Object.defineProperty(this, 'isbn',
    {
        get:function()
        {
            return _isbn;
        },
        set:function(value)
        {
            _isbn = value;
        }
    });

    Object.defineProperty(this, 'title',
    {
        get:function()
        {
            return _title;
        },
        set:function(value)
        {
            _title= value;
        }
    });

    Object.defineProperty(this, 'author',
    {
        get:function()
        {
            return _author;
        },
        set:function(value)
```

```
        {
            _author = value;
        }
    });

    Object.defineProperty(this, 'publicationDate',
    {
        get:function()
        {
            return _publicationDate;
        },
        set:function(value)
        {
            _publicationDate = value;
        }
    });

    Object.defineProperty(this, 'price',
    {
        get:function()
        {
            return _price;
        },
        set:function(value)
        {
            _price = value;
        }
    });
}

Book.prototype.constructor = Book;
```

Mayorga_Usero_JuanJose_DWEC04_hojadeestilo.css

```
/* Cambio la fuente por defecto y aplico un color al fondo. */
body
{
    font-family: Helvetica, sans-serif;
    background-color: cadetblue;
}

/* Estilo de la cabecera: Centrada, con borde redondeado. */
.cabecera
{
    width: 80%;
    margin: auto;
    border-style:solid;
    border-width:5px;
    border-color:black;
    background-color:black;
    color:white;
    border-radius:20px;
    text-align: center;
}

/* Estilo del eneunciado: Centrada, con borde redondeado.*/
.enunciado
{
    width: 80%;
    margin: auto;
    padding: 10px;
    border-style:solid;
```

```
    border-width: 5px;
    border-color: navy;
    background-color: powderblue;
    border-radius:20px;
}

/* Estilo del ejercicio: Centrada, con borde redondeado. */
.ejercicio
{
    width: 80%;
    margin: auto;
    padding: 10px;
    border-style:solid;
    border-width: 5px;
    border-color: navy;
    background-color: powderblue;
    border-radius:20px;
}

.filacabeza
{
    background-color:#000066;
    color:white;
}

.fila:nth-child(odd)
{
    background-color: #b3b3ff;
}
.fila:nth-child(even)
{
    background-color: #e6e6ff;
}
```

Excepciones.js

```
/*
Juan José Mayorga Usero
CFGS Desarrollo de Aplicaciones Web
I.E.S. Maestre de Calatrava - Ciudad Real
Desarrollo Web en Entorno Cliente
UT04 Práctica 1
*/

function BaseException(message = "Default Message", fileName, lineNumber)
{
    let instance = new Error(message, fileName, lineNumber);
    instance.name = "MyError";
    Object.setPrototypeOf(instance, Object.getPrototypeOf(this));
    if(Error.captureStackTrace)
    {
        Error.captureStackTrace(instance, BaseException);
    }
    return instance;
}
BaseException.prototype = Object.create(Error.prototype,
{
    constructor:
    {
        value: BaseException,
        enumerable: false,
        writable: true,
        configurable: true
```

```
    }
});

// Esta excepción nos dice si hemos intentado trabajar con un elemento que no es de la clas
e Book.
function ErrorNoEsBook()
{
    let instance = BaseException.call(this, "El elemento no es un Book.");
    instance.name = "ErrorNoEsBook";
    return instance;
}
ErrorNoEsBook.prototype = Object.create(BaseException.prototype);
ErrorNoEsBook.prototype.constructor = ErrorNoEsBook;

function ErrorListaLlena()
{
    let instance = BaseException.call(this, "La lista está llena. No puedes añadir más elem
entos.");
    instance.name = "ErrorListaLlena";
    return instance;
}
ErrorListaLlena.prototype = Object.create(BaseException.prototype);
ErrorListaLlena.prototype.constructor = ErrorListaLlena;

function ErrorIndiceFuera()
{
    let instance = BaseException.call(this, "El índice está fuera de los elementos de la li
sta.");
    instance.name = "ErrorIndiceFuera";
    return instance;
}
ErrorIndiceFuera.prototype = Object.create(BaseException.prototype);
ErrorIndiceFuera.prototype.constructor = ErrorIndiceFuera;

function ErrorListaVacia()
{
    let instance = BaseException.call(this, "La lista está vacía.");
    instance.name = "ErrorListaVacia";
    return instance;
}
ErrorListaVacia.prototype = Object.create(BaseException.prototype);
ErrorListaVacia.prototype.constructor = ErrorListaVacia;

// Para las listas ordenadas hay varias funciones que no tienen sentido porque se rompería
el orden.
// Como las listas ordenadas heredan de la lista, lo que he hecho ha sido sobreescribir los
 métodos conflictivos de manera que, al invocarlos, se nos diga que esa función está deshab
ilitada.
function FuncionDeshabilitada()
{
    let instance = BaseException.call(this, "Esta función está deshabilitada para las lista
s ordenadas.");
    instance.name = "FuncionDeshabilitada";
    return instance;
}
FuncionDeshabilitada.prototype = Object.create(BaseException.prototype);
FuncionDeshabilitada.prototype.constructor = FuncionDeshabilitada;
```

Ahora muestro cada uno de los bloques para Lista, ListaOrdenadaPorISBN y ListaOrdenadaPorFecha.

Lista.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>DWEC04 - Práctica 1 - Juan José Mayorga Usero</title>
        <!--
 Utilizo una hoja de estilos que he creado para mejorar la presentación de la Tarea. -->
        <link  rel="stylesheet" type="text/css"  href="Mayorga_Usero_JuanJose_DWEC04_hojade
estilo.css"/>
    </head>
    <body>
        <!-- Creo un div con mis datos y los del módulo, al que aplico un estilo. -->
        <div class="cabecera">
            <h3>CFGS Desarrollo de Aplicaciones Web</h3>
            <h3>I.E.S. Maestre de Calatrava - Ciudad Real</h3>
            <h3>Desarrollo Web en Entorno Cliente</h3>
            <h3>Juan José Mayorga Usero</h3>
        </div>
        <br/>
        <!-- Creo un div con el enunciado del apartado, al que aplico un estilo. -->
        <div class="enunciado">
            <h1>DWEC04</h1>
            <h2>UT04 Práctica 1</h2>
            <h3>1. Listas <br/>
                Dada la práctica de la UT03.1 para la implementación de Listas y ListasOrde
nadas, rediseña el código para que estén implementadas mediante objetos para facilitar su m
antenimiento.<br/>
                Deberás implementar un conjunto de excepciones en forma de objeto para comu
nicar errores. Los objetos para representar las excepciones deben ser específicos de la prá
ctica, no errores genéricos.<br/>
                Para los objetos de ListasOrdenadas deben heredar de Lista los métodos que
sean iguales, y sobrescribir los que personalicen la funcionalidad.<br/>
                Haz uso de los recursos que consideres para implementar la práctica, propie
dades públicas, campos privados, métodos públicos, si necesitas getter o setter, propiedade
s o métodos estáticos. No será necesario utilizarlos todos, ni habrá un mínimo de ellos que
 sean necesario, tan solo que se utilicen con coherencia.
            </h3>
        </div>
        <br/>
        <!-- En este div resuelvo el ejercicio. -->
        <div class="ejercicio">
            <h3>Lista</h3>
            <p>Para comprobar el correcto funcionamiento de cada una de las funciones imple
mentadas, he desarollado una función de testeo en un archivo .js independiente en el que se
 deja un registro de cada una de las operación a través de la consola. <br/>
            Para acceder a la consola:<br/>
            - En Google Chrome, pulsar Ctrl + Shift + j.<br/>
            - En Mozilla Firefox, pulsar Ctrl + Shift + k.<br/>
            - En Opera, pulsar Ctrl + Shift + j.<br/>
            - En Microsoft Edge, pulsar Ctrl + Shift + j.
            <br/>
            </p>
        </div>
        <!-- Indico la ruta del código javascript. -->
        <script src="Book.js"></script>
        <script src="Excepciones.js"></script>
        <script src="Lista.js"></script>
        <script src="Lista-test.js"></script>
    </body>
</html>
```

Lista.js

```
/*
Juan José Mayorga Usero
```

```
CFGS Desarrollo de Aplicaciones Web
I.E.S. Maestre de Calatrava - Ciudad Real
Desarrollo Web en Entorno Cliente
UT04 Práctica 1
*/

/* Creo el constructor para los objetos Lista. Recibe como argumento de entrada el número d
e elementos que puede albergar la lista. */
function Lista(max_elem_lista)
{
    let _max_elem_lista = max_elem_lista;
    // La lista contiene un array de books que se irá actualizando con las operaciones que
se hacen.
    let _books = [];

    Object.defineProperty(this, 'max_elem_lista',
    {
        get:function()
        {
            return _max_elem_lista;
        },
        set:function(value)
        {
            _max_elem_lista = value;
        }
    });

    Object.defineProperty(this, 'books',
    {
        get:function()
        {
            return _books;
        },
    });
}

Lista.prototype.constructor=Lista;

// Devuelve true o false en función de si la lista está vacía.
Lista.prototype.isEmpty = function()
{
    return (this.books.length === 0);
}

// Devuelve true o false en función de si la lista está llena.
Lista.prototype.isFull = function()
{
    return (this.books.length === this.max_elem_lista);
}

// Devuelve el número de elementos de la lista.
Lista.prototype.size = function()
{
    return (this.books.length);
}

// Añade un nuevo elemento al final de la lista. Devuelve el tamaño de la lista una vez aña
dido.
Lista.prototype.add = function(elem)
{
    try
    {
        // Excepción: El elemento no es un Book. Puedes comprobar si tiene la propiedad ISB
N y title.
```

```
        // Como en este caso he creado también los objetos Book, podría usar elem.isbn para
 acceder a la propiedad isbn.
        if (elem.hasOwnProperty('isbn')==false||elem.hasOwnProperty('title')==false) // htt
ps://dmitripavlutin.com/check-if-object-has-property-javascript/
        {
            throw new ErrorNoEsBook();
        }
        // Excepción: La lista está llena.
        else if (this.isFull())
        {
            throw new ErrorListaLlena(); // Si está completa lanzamos excepción.
        }
        // Añadimos si la lista no está completa.
        this.books.push(elem); //Utilizamos los métodos de array para gestionar la lista.
        return this.size(); // Devolvemos el tamaño.
    }
    catch(error)
    {
        console.log(error.toString());
    }
}

// Añade un nuevo elemento en la posición especificada en la lista. Devuelve el tamaño de l
a lista una vez añadido.
Lista.prototype.addAt = function(elem, index)
{
    try
    {
        // Excepción: El elemento no es un Book. Puedes comprobar si tiene la propiedad ISB
N y title.
        if (elem.hasOwnProperty('isbn')==false||elem.hasOwnProperty('title')==false) // htt
ps://dmitripavlutin.com/check-if-object-has-property-javascript/
        {
            throw new ErrorNoEsBook();
        }
        // Excepción: La lista está llena.
        else if (this.isFull())
        {
            throw new ErrorListaLlena(); // Si está completa lanzamos excepción.
        }
        // Excepción: El índice está fuera de los elementos de la lista.
        else if(index>=this.max_elem_lista||index<0)
        {
            throw new ErrorIndiceFuera();
        }
        // Añadimos si la lista no está completa.
        this.books.splice(index, 0, elem); //Utilizamos los métodos de array para gestionar
 la lista.
        return this.size(); // Devolvemos el tamaño.
    }
    catch(error)
    {
        console.log(error.toString());
    }
}

// Devuelve el elemento de la lista de la posición especificada.
Lista.prototype.get = function(index)
{
    try
    {
        // Excepción: El índice está fuera de los elementos de la lista.
        if(index>=this.max_elem_lista||index<0)
        {
            throw new ErrorIndiceFuera();
```

```
        }
        return this.books[index];
    }
    catch(error)
    {
        console.log(error.toString());
    }
}

// Devuelve la lista en formato cadena. El delimitador de elementos será " - ".
/* Dado que se trata de un objeto con múltiples propiedades, he considerado más claro repre
sentar la lista mostrando las propiedades de cada objeto en una línea, y dejar un espacio e
ntre cada objeto. */
Lista.prototype.toString = function()
{
    // Reducimos el array para el primer elemento en salir.
    // https://www.w3schools.com/jsref/jsref_reduce.asp
    return this.books.reduce(function(str, item, index)
    {
        return str + item.isbn + "\n" + item.title +"\n" + item.author +"\n" + item.publica
tionDate.getFullYear() + "\n" +item.price  + "\n\n";
    },""); //El valor inicial de la reducción es ""
}

// Devuelve la posición del elemento indicado. Si el elemento no está en la lista devuelve
-1. Realiza la comparación por ISBN.
Lista.prototype.indexOf = function(elem)
{
    try
    {
        // Excepción: El elemento no es un Book. Puedes comprobar si tiene la propiedad ISB
N y title.
        if (elem.hasOwnProperty('isbn')==false||elem.hasOwnProperty('title')==false) // htt
ps://dmitripavlutin.com/check-if-object-has-property-javascript/
        {
            throw new ErrorNoEsBook();
        }
        // https://www.geeksforgeeks.org/find-start-ending-index-element-unsorted-array/
        // https://dmitripavlutin.com/how-to-compare-objects-in-javascript/
        var index = -1;
        // Recorrer la lista de principio a fin para encontrar la primera coincidencia.
        for (var i = 0; i < this.books.length; i++)
        {
            if (this.books[i].isbn == elem.isbn)
            {
                index = i;
                break;
            }
        }
        return index;
    }
    catch(error)
    {
        console.log(error.toString());
    }
}

// Devuelve la posición del elemento indicado. Si el elemento no está en la lista devuelve
-1. Realiza la comparación por ISBN.
Lista.prototype.lastIndexOf = function(elem)
{
    try
    {
        // Excepción: El elemento no es un Book. Puedes comprobar si tiene la propiedad ISB
N y title.
```

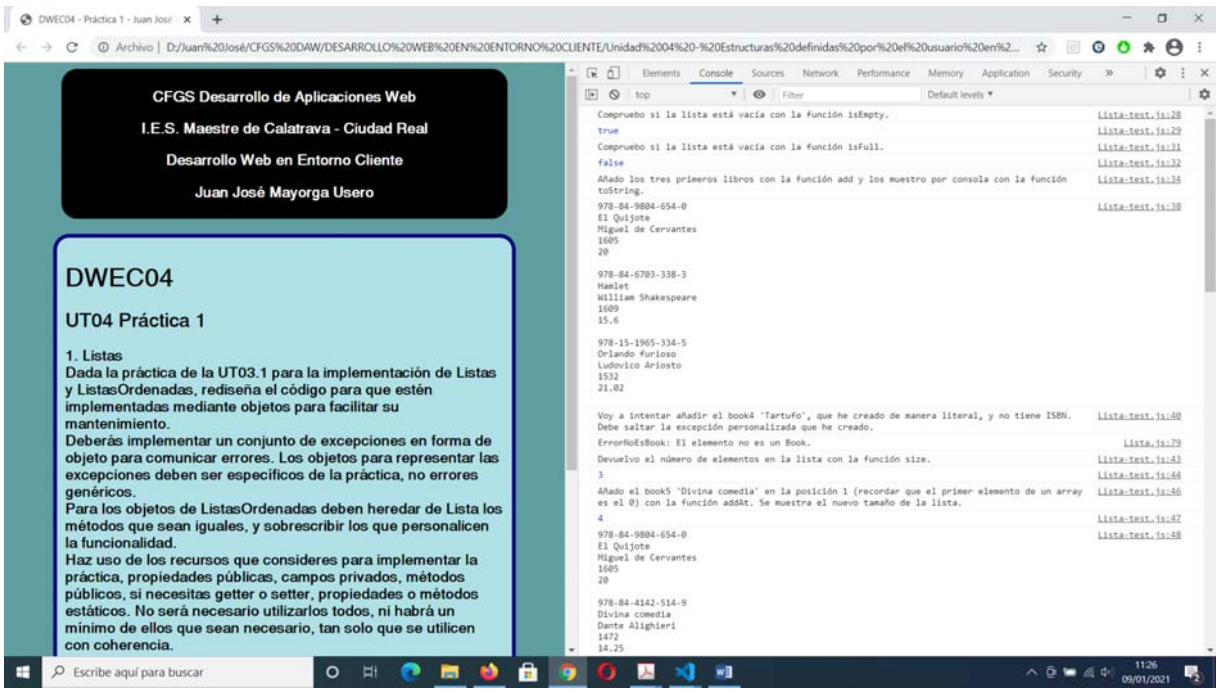
```
        if (elem.hasOwnProperty('isbn')==false||elem.hasOwnProperty('title')==false) // htt
ps://dmitripavlutin.com/check-if-object-has-property-javascript/
        {
            throw new ErrorNoEsBook();
        }
        var index = -1;
        // Recorrer la lista de fin a principio para encontrar la primera coincidencia.
        for (var i = this.books.length-1; i >=0; i--)
        {
            if (this.books[i].isbn == elem.isbn)
            {
                index = i;
                break;
            }
        }
        return index;
    }
    catch(error)
    {
        console.log(error.toString());
    }
}

// Devuelve el máximo número de elementos que podemos tener en la lista.
Lista.prototype.capacity = function()
{
    return this.max_elem_lista;
}

// Vacía la lista.
Lista.prototype.clear = function()
{
    this.books.splice(0, this.books.length); // https://www.w3schools.com/jsref/jsref_splic
e.asp
}

// Devuelve el primer elemento de la lista.
Lista.prototype.firstElement = function()
{
    try
    {
        if (this.isEmpty())
        {
            throw new ErrorListaVacia();
        }
        return this.books[0];
    }
    catch(error)
    {
        console.log(error.toString());
    }
}

// Devuelve el último elemento de la lista.
Lista.prototype.lastElement = function()
{
    try
    {
        // Excepción: La lista está vacía.
        if (this.isEmpty())
        {
            throw new ErrorListaVacia();
        }
        return this.books[this.books.length-1];
    }
```

```
    catch(error)
    {
        console.log(error.toString());
    }
}

// Elimina el elemento de la posición indicada. Devuelve el elemento borrado.
Lista.prototype.remove = function(index)
{
    try
    {
        // Excepción: El índice está fuera de los elementos de la lista.
        if(index>=this.max_elem_lista||index<0)
        {
            throw new ErrorIndiceFuera();
        }
        return this.books.splice(index, 1)[0]; // https://www.w3schools.com/jsref/jsref_spl
ice.asp
        // Hay que poner el índice 0, porque si no, devuelve un array de un elemento. Quere
mos el primer elemento de ese array.
    }
    catch(error)
    {
        console.log(error.toString());
    }
}

// Elimina el elemento indicado de la lista. Devuelve true si se ha podido borrar el elemen
to, false en caso contrario.
Lista.prototype.removeElement = function(elem)
{
    return this.remove(this.indexOf(elem));
}

// Reemplaza el elemento de la lista indicado por el índice. Devuelve el elemento que estab
a anteriormente en la lista.
Lista.prototype.set = function(elem, index)
{
    try
    {
        // Excepción: El elemento no es un Book. Puedes comprobar si tiene la propiedad ISB
N y title.
        if (elem.hasOwnProperty('isbn')==false||elem.hasOwnProperty('title')==false) // htt
ps://dmitripavlutin.com/check-if-object-has-property-javascript/
        {
            throw new ErrorNoEsBook();
        }
        // Excepción: El índice está fuera de los elementos de la lista.
        else if(index>=this.max_elem_lista||index<0)
        {
            throw new ErrorIndiceFuera();
        }
        // https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales
/Array/splice
        return this.books.splice(index,1,elem)[0]; //Utilizamos los métodos de array para g
estionar la lista
        // Hay que poner el índice 0, porque si no, devuelve un array de un elemento. Quere
mos el primer elemento de ese array.
    }
    catch(error)
    {
        console.log(error.toString());
    }
}
```

Lista-test.js

```
/*
Juan José Mayorga Usero
CFGS Desarrollo de Aplicaciones Web
I.E.S. Maestre de Calatrava - Ciudad Real
Desarrollo Web en Entorno Cliente
UT04 Práctica 1
*/

// Función de testeo.
function testList()
{
    // Primero creo unos cuantos objetos Book para probar los métodos de la Lista que crear
é después.
    let book1=new Book("978-84-9804-654-
0", "El Quijote", "Miguel de Cervantes", new Date(1605, 0, 1), 20);
    let book2=new Book("978-84-6703-338-
3", "Hamlet", "William Shakespeare", new Date(1609, 0, 1), 15.6);
    let book3=new Book("978-15-1965-334-
5", "Orlando furioso", "Ludovico Ariosto", new Date(1532, 0, 1), 21.02);
    let book4 = {
    title: "Tartufo",
    author: "Molière",
    publicationDate:
    new Date(1669, 0, 1),
    price: 9.17, };
    let book5=new Book("978-84-4142-514-
9", "Divina comedia", "Dante Alighieri", new Date(1472, 0, 1), 14.25);
    let book6=new Book("978-84-6703-458-
5", "El perro del hortelano", "Lope de Vega", new Date(1618, 0, 1), 6.60);
    let book7=new Book("978-95-0039-253-
2", "Leviathan", "Thomas Hobbes", new Date(1651, 0, 1), 6.60);
    // Ahora creo una lista de con un número máximo de ítems de 5.
    lista=new Lista(5);
    // Compruebo si la lista está vacía con la función isEmpty.
    console.log("Compruebo si la lista está vacía con la función isEmpty.");
    console.log(lista.isEmpty());
    // Compruebo si la lista está llena con la función isFull.
    console.log("Compruebo si la lista está vacía con la función isFull.");
    console.log(lista.isFull());
    // Añado los tres primeros libros con la función add y los muestro por consola con la f
unción toString.
    console.log("Añado los tres primeros libros con la función add y los muestro por consol
a con la función toString.");
    lista.add(book1);
    lista.add(book2);
    lista.add(book3);
    console.log(lista.toString());
    // Voy a intentar añadir el book4 'Tartufo', que he creado de manera literal, y no tien
e ISBN. Debe saltar la excepción personalizada que he creado.
    console.log("Voy a intentar añadir el book4 'Tartufo', que he creado de manera literal,
 y no tiene ISBN. Debe saltar la excepción personalizada que he creado.");
    lista.add(book4);
    // Devuelvo el número de elementos en la lista con la función size.
    console.log("Devuelvo el número de elementos en la lista con la función size.");
    console.log(lista.size());
    // Añado el book5 'Divina comedia' en la posición 1 (recordar que el primer elemento de
 un array es el 0) con la función addAt. Se muestra el nuevo tamaño de la lista.
    console.log("Añado el book5 'Divina comedia' en la posición 1 (recordar que el primer e
lemento de un array es el 0) con la función addAt. Se muestra el nuevo tamaño de la lista."
);
  console.log(lista.addAt(book5, 1));
    console.log(lista.toString());
    // Obtengo el segundo elemento de la lista con el método get.
```

```
    console.log("Obtengo el segundo elemento de la lista con el método get.");
    console.log(lista.get(1));
    // Hallo el índice del objeto book3 'Orlando furioso', con la función indexOf.
    console.log("Hallo el índice del objeto book3 'Orlando furioso', con la función indexOf
.");
    console.log(lista.indexOf(book3));
    // Intento hallar el índice del objeto book6 'El perro del hortelano', que no está en l
a lista.
    console.log("Intento hallar el índice del objeto book6 'El perro del hortelano', que no
 está en la lista.");
    console.log(lista.indexOf(book6));
    // Hallo el índice del objeto book3 'Orlando furioso', comenzando a buscar por el final
, con la función lastIndexOf.
    console.log("Hallo el índice del objeto book3 'Orlando furioso', comenzando a buscar po
r el final, con la función lastIndexOf.");
    console.log(lista.lastIndexOf(book3));
    // Muestro el máximo número de elementos que podemos tener en la lista con la función c
apacity.
    console.log("Muestro el máximo número de elementos que podemos tener en la lista con la
 función capacity.");
    console.log(lista.capacity());
    // Obtengo el primer elemento de la lista con la función firstElement.
    console.log("Obtengo el primer elemento de la lista con la función firstElement.");
    console.log(lista.firstElement());
    // Obtengo el último elemento de la lista con la función firstElement.
    console.log("Obtengo el último elemento de la lista con la función firstElement.");
    console.log(lista.lastElement());
    // Elimino el elemento de la lista que está en la posición 2 (recordar que el primer el
emento de un array es el 0) con la función remove.
    console.log("Elimino el elemento de la lista que está en la posición 2 (recordar que el
 primer elemento de un array es el 0) con la función remove.");
    console.log(lista.remove(2));
    // Muestro cómo queda la lista tras esta operación con la función toString.
    console.log("Muestro cómo queda la lista tras esta operación con la función toString.")
;
    console.log(lista.toString());
    // Elimino el elemento book3 'Orlando furioso' con la función removeElement.
    console.log("Elimino el elemento book3 'Orlando furioso' con la función removeElement."
);
    console.log(lista.removeElement(book3));
    // Muestro cómo queda la lista tras esta operación con la función toString.
    console.log("Muestro cómo queda la lista tras esta operación con la función toString.")
;
    console.log(lista.toString());
    // Ahora sustituyo el elemento en la posición 0 (recordar que el primer elemento de un
array es el 0) por el objeto book6 'El perro del hortelano'.
    console.log("Ahora sustituyo el elemento en la posición 0 (recordar que el primer eleme
nto de un array es el 0) por el objeto book6 'El perro del hortelano'.");
    console.log(lista.set(book6, 0));
    // Muestro cómo queda la lista tras esta operación con la función toString.
    console.log("Muestro cómo queda la lista tras esta operación con la función toString.")
;
    console.log(lista.toString());
    // Vacío la lista con la función clear.
    console.log("Vacío la lista con la función clear.");
    lista.clear();
    // Muestro cómo queda la lista tras esta operación con la función toString.
    console.log("Muestro cómo queda la lista tras esta operación con la función toString.")
;
    console.log(lista.toString());
}
window.onload = testList();
```



ListaOrdenadaPorISBN.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>DWEC04 - Práctica 1 - Juan José Mayorga Usero</title>
        <!--
 Utilizo una hoja de estilos que he creado para mejorar la presentación de la Tarea. -->
        <link  rel="stylesheet" type="text/css"  href="Mayorga_Usero_JuanJose_DWEC04_hojade
estilo.css"/>
    </head>
    <body>
        <!-- Creo un div con mis datos y los del módulo, al que aplico un estilo. -->
        <div class="cabecera">
            <h3>CFGS Desarrollo de Aplicaciones Web</h3>
            <h3>I.E.S. Maestre de Calatrava - Ciudad Real</h3>
            <h3>Desarrollo Web en Entorno Cliente</h3>
            <h3>Juan José Mayorga Usero</h3>
        </div>
        <br/>
        <!-- Creo un div con el enunciado del apartado, al que aplico un estilo. -->
        <div class="enunciado">
            <h1>DWEC04</h1>
            <h2>UT04 Práctica 1</h2>
            <h3>1. Listas <br/>
                Dada la práctica de la UT03.1 para la implementación de Listas y ListasOrde
nadas, rediseña el código para que estén implementadas mediante objetos para facilitar su m
antenimiento.<br/>
                Deberás implementar un conjunto de excepciones en forma de objeto para comu
nicar errores. Los objetos para representar las excepciones deben ser específicos de la prá
ctica, no errores genéricos.<br/>
                Para los objetos de ListasOrdenadas deben heredar de Lista los métodos que
sean iguales, y sobrescribir los que personalicen la funcionalidad.<br/>
                Haz uso de los recursos que consideres para implementar la práctica, propie
dades públicas, campos privados, métodos públicos, si necesitas getter o setter, propiedade
s o métodos estáticos. No será necesario utilizarlos todos, ni habrá un mínimo de ellos que
 sean necesario, tan solo que se utilicen con coherencia.
```

```
            </h3>
        </div>
        <br/>
        <!-- En este div resuelvo el ejercicio. -->
        <div class="ejercicio">
            <h3>Lista ordenada por ISBN</h3>
            <p>Para comprobar el correcto funcionamiento de cada una de las funciones imple
mentadas, he desarollado una función de testeo en un archivo .js independiente en el que se
 deja un registro de cada una de las operación a través de la consola. <br/>
            Para acceder a la consola:<br/>
            - En Google Chrome, pulsar Ctrl + Shift + j.<br/>
            - En Mozilla Firefox, pulsar Ctrl + Shift + k.<br/>
            - En Opera, pulsar Ctrl + Shift + j.<br/>
            - En Microsoft Edge, pulsar Ctrl + Shift + j.
            <br/>
            </p>
        </div>
        <!-- Indico la ruta del código javascript. -->
        <script src="Book.js"></script>
        <script src="Excepciones.js"></script>
        <script src="Lista.js"></script>
        <script src="ListaOrdenadaPorISBN.js"></script>
        <script src="ListaOrdenadaPorISBN-test.js"></script>
    </body>
</html>
```

ListaOrdenadaPorISBN.js

```
/*
Juan José Mayorga Usero
CFGS Desarrollo de Aplicaciones Web
I.E.S. Maestre de Calatrava - Ciudad Real
Desarrollo Web en Entorno Cliente
UT04 Práctica 1
*/

// Hago que ListaOrdenadaPorISBN herede de Lista.
ListaOrdenadaPorISBN.prototype = Object.create(Lista.prototype);

ListaOrdenadaPorISBN.prototype.constructor = ListaOrdenadaPorISBN;

// Igual que Lista, el constructor sólo recibe como argumento el número de elementos admisi
bles.
function ListaOrdenadaPorISBN(max_elem_lista)
{
    Lista.call(this, max_elem_lista);
}

// Añade un nuevo elemento a la lista manteniendo la relación de orden. Devuelve el tamaño
de la lista una vez añadido.
ListaOrdenadaPorISBN.prototype.add = function(elem)
{
    try
    {
        // Excepción: El elemento no es un Book. Puedes comprobar si tiene la propiedad ISB
N y title.
        if (elem.hasOwnProperty('isbn')==false||elem.hasOwnProperty('title')==false) // htt
ps://dmitripavlutin.com/check-if-object-has-property-javascript/
        {
            throw new ErrorNoEsBook();
        }
        // Excepción: La lista está llena.
        else if (this.isFull())
        {
```

```
            throw new ErrorListaLlena(); // Si está completa lanzamos excepción.
        }
        // Añadimos si la lista no está completa.
        this.books.push(elem); //Utilizamos los métodos de array para gestionar la lista.
        this.books.sort(compare);
        return this.size(); // Devolvemos el tamaño.
    }
    catch(error)
    {
        console.log(error.toString());
    }
}

// Función para ordenar el array por fecha de publicación de manera ascendente.
function compare(a,b) // https://stackoverflow.com/questions/1129216/sort-array-of-objects-
by-string-property-value
{
    if(a.isbn<b.isbn)
    {
        return -1;
    }
    if(a.isbn>b.isbn)
    {
        return 1;
    }
    return 0;
}

/*
Para las funciones que no tienen sentido en listas ordenadas:
- addAt
- lastIndexOf
- set
he optado por sobreescribirlas y lanzar una excepción que indique que se está intentando ut
ilizar una función deshabilitada.
*/
ListaOrdenadaPorISBN.prototype.addAt = function(elem, index)
{
    try
    {
        throw new FuncionDeshabilitada();
    }
    catch(error)
    {
        console.log(error.toString());
    }
}

ListaOrdenadaPorISBN.prototype.lastIndexOf = function(elem)
{
    try
    {
        throw new FuncionDeshabilitada();
    }
    catch(error)
    {
        console.log(error.toString());
    }
}

ListaOrdenadaPorISBN.prototype.set = function(elem, index)
{
    try
    {
        throw new FuncionDeshabilitada();
```

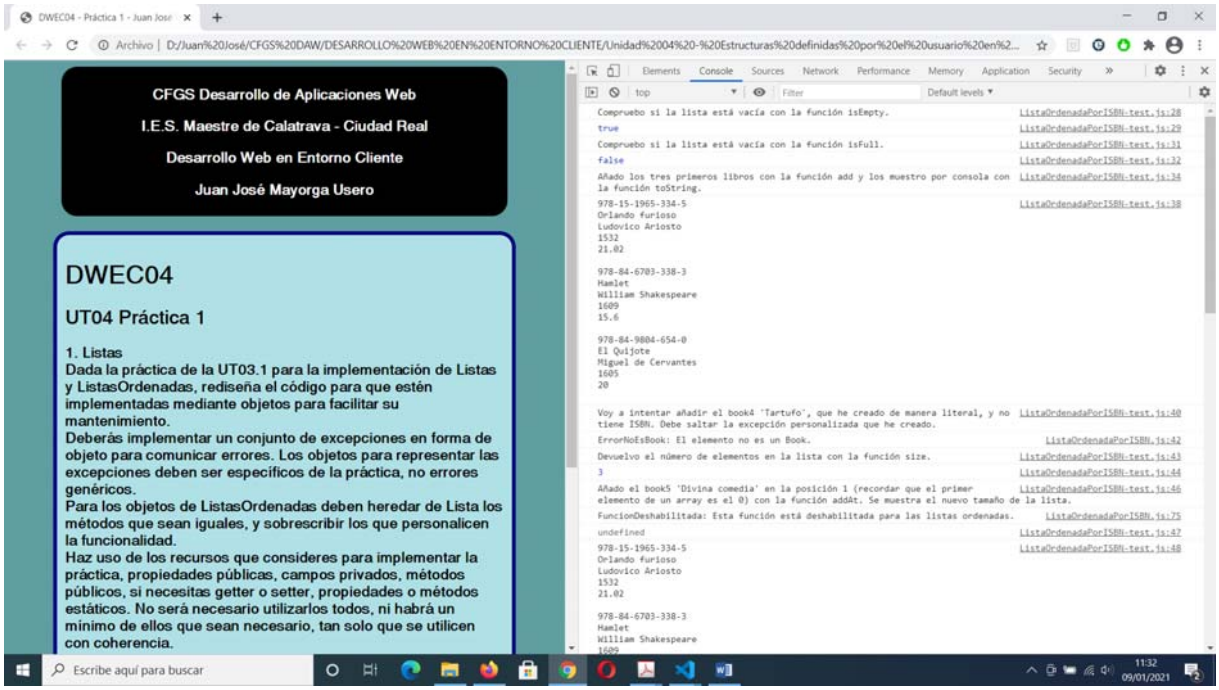
```
    }
    catch(error)
    {
        console.log(error.toString());
    }
}
```

ListaOrdenadaPorISBN-test.js

```
/*
Juan José Mayorga Usero
CFGS Desarrollo de Aplicaciones Web
I.E.S. Maestre de Calatrava - Ciudad Real
Desarrollo Web en Entorno Cliente
UT04 Práctica 1
*/

// Función de testeo.
function testList()
{
    // Primero creo unos cuantos objetos Book para probar los métodos de la Lista que crear
é después.
    let book1=new Book("978-84-9804-654-
0", "El Quijote", "Miguel de Cervantes", new Date(1605, 0, 1), 20);
    let book2=new Book("978-84-6703-338-
3", "Hamlet", "William Shakespeare", new Date(1609, 0, 1), 15.6);
    let book3=new Book("978-15-1965-334-
5", "Orlando furioso", "Ludovico Ariosto", new Date(1532, 0, 1), 21.02);
    let book4 = {
    title: "Tartufo",
    author: "Molière",
    publicationDate:
    new Date(1669, 0, 1),
    price: 9.17, };
    let book5=new Book("978-84-4142-514-
9", "Divina comedia", "Dante Alighieri", new Date(1472, 0, 1), 14.25);
    let book6=new Book("978-84-6703-458-
5", "El perro del hortelano", "Lope de Vega", new Date(1618, 0, 1), 6.60);
    let book7=new Book("978-95-0039-253-
2", "Leviathan", "Thomas Hobbes", new Date(1651, 0, 1), 6.60);
    // Ahora creo una lista de con un número máximo de ítems de 5.
    lista=new ListaOrdenadaPorISBN(5);
    // Compruebo si la lista está vacía con la función isEmpty.
    console.log("Compruebo si la lista está vacía con la función isEmpty.");
    console.log(lista.isEmpty());
    // Compruebo si la lista está llena con la función isFull.
    console.log("Compruebo si la lista está vacía con la función isFull.");
    console.log(lista.isFull());
    // Añado los tres primeros libros con la función add y los muestro por consola con la f
unción toString.
    console.log("Añado los tres primeros libros con la función add y los muestro por consol
a con la función toString.");
    lista.add(book1);
    lista.add(book2);
    lista.add(book3);
    console.log(lista.toString());
    // Voy a intentar añadir el book4 'Tartufo', que he creado de manera literal, y no tien
e ISBN. Debe saltar la excepción personalizada que he creado.
    console.log("Voy a intentar añadir el book4 'Tartufo', que he creado de manera literal,
 y no tiene ISBN. Debe saltar la excepción personalizada que he creado.");
    lista.add(book4);
    // Devuelvo el número de elementos en la lista con la función size.
    console.log("Devuelvo el número de elementos en la lista con la función size.");
    console.log(lista.size());
```

```
    // Añado el book5 'Divina comedia' en la posición 1 (recordar que el primer elemento de
 un array es el 0) con la función addAt. Se muestra el nuevo tamaño de la lista.
    console.log("Añado el book5 'Divina comedia' en la posición 1 (recordar que el primer e
lemento de un array es el 0) con la función addAt. Se muestra el nuevo tamaño de la lista."
);
  console.log(lista.addAt(book5, 1));
    console.log(lista.toString());
    // Obtengo el segundo elemento de la lista con el método get.
    console.log("Obtengo el segundo elemento de la lista con el método get.");
    console.log(lista.get(1));
    // Hallo el índice del objeto book3 'Orlando furioso', con la función indexOf.
    console.log("Hallo el índice del objeto book3 'Orlando furioso', con la función indexOf
.");
    console.log(lista.indexOf(book3));
    // Intento hallar el índice del objeto book6 'El perro del hortelano', que no está en l
a lista.
    console.log("Intento hallar el índice del objeto book6 'El perro del hortelano', que no
 está en la lista.");
    console.log(lista.indexOf(book6));
    // Hallo el índice del objeto book3 'Orlando furioso', comenzando a buscar por el final
, con la función lastIndexOf.
    console.log("Hallo el índice del objeto book3 'Orlando furioso', comenzando a buscar po
r el final, con la función lastIndexOf.");
    console.log(lista.lastIndexOf(book3));
    // Muestro el máximo número de elementos que podemos tener en la lista con la función c
apacity.
    console.log("Muestro el máximo número de elementos que podemos tener en la lista con la
 función capacity.");
    console.log(lista.capacity());
    // Obtengo el primer elemento de la lista con la función firstElement.
    console.log("Obtengo el primer elemento de la lista con la función firstElement.");
    console.log(lista.firstElement());
    // Obtengo el último elemento de la lista con la función firstElement.
    console.log("Obtengo el último elemento de la lista con la función firstElement.");
    console.log(lista.lastElement());
    // Elimino el elemento de la lista que está en la posición 2 (recordar que el primer el
emento de un array es el 0) con la función remove.
    console.log("Elimino el elemento de la lista que está en la posición 2 (recordar que el
 primer elemento de un array es el 0) con la función remove.");
    console.log(lista.remove(2));
    // Muestro cómo queda la lista tras esta operación con la función toString.
    console.log("Muestro cómo queda la lista tras esta operación con la función toString.")
;
    console.log(lista.toString());
    // Elimino el elemento book3 'Orlando furioso' con la función removeElement.
    console.log("Elimino el elemento book3 'Orlando furioso' con la función removeElement."
);
    console.log(lista.removeElement(book3));
    // Muestro cómo queda la lista tras esta operación con la función toString.
    console.log("Muestro cómo queda la lista tras esta operación con la función toString.")
;
    console.log(lista.toString());
    // Ahora sustituyo el elemento en la posición 0 (recordar que el primer elemento de un
array es el 0) por el objeto book6 'El perro del hortelano'.
    console.log("Ahora sustituyo el elemento en la posición 0 (recordar que el primer eleme
nto de un array es el 0) por el objeto book6 'El perro del hortelano'.");
    console.log(lista.set(book6, 0));
    // Muestro cómo queda la lista tras esta operación con la función toString.
    console.log("Muestro cómo queda la lista tras esta operación con la función toString.")
;
    console.log(lista.toString());
    // Vacío la lista con la función clear.
    console.log("Vacío la lista con la función clear.");
    lista.clear();
    // Muestro cómo queda la lista tras esta operación con la función toString.
```

```
    console.log("Muestro cómo queda la lista tras esta operación con la función toString.")
;
    console.log(lista.toString());
}
window.onload = testList();
```



ListaOrdenadaPorFecha.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>DWEC04 - Práctica 1 - Juan José Mayorga Usero</title>
        <!--
 Utilizo una hoja de estilos que he creado para mejorar la presentación de la Tarea. -->
        <link  rel="stylesheet" type="text/css"  href="Mayorga_Usero_JuanJose_DWEC04_hojade
estilo.css"/>
    </head>
    <body>
        <!-- Creo un div con mis datos y los del módulo, al que aplico un estilo. -->
        <div class="cabecera">
            <h3>CFGS Desarrollo de Aplicaciones Web</h3>
            <h3>I.E.S. Maestre de Calatrava - Ciudad Real</h3>
            <h3>Desarrollo Web en Entorno Cliente</h3>
            <h3>Juan José Mayorga Usero</h3>
        </div>
        <br/>
        <!-- Creo un div con el enunciado del apartado, al que aplico un estilo. -->
        <div class="enunciado">
            <h1>DWEC04</h1>
            <h2>UT04 Práctica 1</h2>
            <h3>1. Listas <br/>
                Dada la práctica de la UT03.1 para la implementación de Listas y ListasOrde
nadas, rediseña el código para que estén implementadas mediante objetos para facilitar su m
antenimiento.<br/>
```

```
                Deberás implementar un conjunto de excepciones en forma de objeto para comu
nicar errores. Los objetos para representar las excepciones deben ser específicos de la prá
ctica, no errores genéricos.<br/>
                Para los objetos de ListasOrdenadas deben heredar de Lista los métodos que
sean iguales, y sobrescribir los que personalicen la funcionalidad.<br/>
                Haz uso de los recursos que consideres para implementar la práctica, propie
dades públicas, campos privados, métodos públicos, si necesitas getter o setter, propiedade
s o métodos estáticos. No será necesario utilizarlos todos, ni habrá un mínimo de ellos que
 sean necesario, tan solo que se utilicen con coherencia.
            </h3>
        </div>
        <br/>
        <!-- En este div resuelvo el ejercicio. -->
        <div class="ejercicio">
            <h3>Lista ordenada por fecha de publicación</h3>
            <p>Para comprobar el correcto funcionamiento de cada una de las funciones imple
mentadas, he desarollado una función de testeo en un archivo .js independiente en el que se
 deja un registro de cada una de las operación a través de la consola. <br/>
            Para acceder a la consola:<br/>
            - En Google Chrome, pulsar Ctrl + Shift + j.<br/>
            - En Mozilla Firefox, pulsar Ctrl + Shift + k.<br/>
            - En Opera, pulsar Ctrl + Shift + j.<br/>
            - En Microsoft Edge, pulsar Ctrl + Shift + j.
            <br/>
            </p>
        </div>
        <!-- Indico la ruta del código javascript. -->
        <script src="Book.js"></script>
        <script src="Excepciones.js"></script>
        <script src="Lista.js"></script>
        <script src="ListaOrdenadaPorFecha.js"></script>
        <script src="ListaOrdenadaPorFecha-test.js"></script>
    </body>
</html>
```

ListaOrdenadaPorFecha.js

```
/*
Juan José Mayorga Usero
CFGS Desarrollo de Aplicaciones Web
I.E.S. Maestre de Calatrava - Ciudad Real
Desarrollo Web en Entorno Cliente
UT04 Práctica 1
*/

// Hago que ListaOrdenadaPorISBN herede de Lista.
ListaOrdenadaPorFecha.prototype = Object.create(Lista.prototype);

ListaOrdenadaPorFecha.prototype.constructor = ListaOrdenadaPorFecha;

// Igual que Lista, el constructor sólo recibe como argumento el número de ele
mentos admisibles.
function ListaOrdenadaPorFecha(max_elem_lista)
{
    Lista.call(this, max_elem_lista);
}

// Añade un nuevo elemento a la lista manteniendo la relación de orden. Devuel
ve el tamaño de la lista una vez añadido.
ListaOrdenadaPorFecha.prototype.add = function(elem)
```

```
{
    try
    {
        // Excepción: El elemento no es un Book. Puedes comprobar si tiene la
propiedad ISBN y title.
        if (elem.hasOwnProperty('isbn')==false||elem.hasOwnProperty('title')==
false) // https://dmitripavlutin.com/check-if-object-has-property-javascript/
        {
            throw new ErrorNoEsBook();
        }
        // Excepción: La lista está llena.
        else if (this.isFull())
        {
            throw new ErrorListaLlena(); // Si está completa lanzamos excepció
n.
        }
        // Añadimos si la lista no está completa.
        this.books.push(elem); //Utilizamos los métodos de array para gestiona
r la lista.
        this.books.sort(compare);
        return this.size(); // Devolvemos el tamaño.
    }
    catch(error)
    {
        console.log(error.toString());
    }
}

// Función para ordenar el array por fecha de publicación de manera ascendente
.
function compare(a,b) // https://stackoverflow.com/questions/1129216/sort-
array-of-objects-by-string-property-value
{
    if(a.publicationDate<b.publicationDate)
    {
        return -1;
    }
    if(a.publicationDate>b.publicationDate)
    {
        return 1;
    }
    return 0;
}

/*
Para las funciones que no tienen sentido en listas ordenadas:
- addAt
- lastIndexOf
- set
he optado por sobreescribirlas y lanzar una excepción que indique que se está
intentando utilizar una función deshabilitada.
*/
ListaOrdenadaPorFecha.prototype.addAt = function(elem, index)
{
    try
    {
```

```
        throw new FuncionDeshabilitada();
    }
    catch(error)
    {
        console.log(error.toString());
    }
}

ListaOrdenadaPorFecha.prototype.lastIndexOf = function(elem)
{
    try
    {
        throw new FuncionDeshabilitada();
    }
    catch(error)
    {
        console.log(error.toString());
    }
}

ListaOrdenadaPorFecha.prototype.set = function(elem, index)
{
    try
    {
        throw new FuncionDeshabilitada();
    }
    catch(error)
    {
        console.log(error.toString());
    }
}
```

ListaOrdenadaPorFecha-test.js

```
/*
Juan José Mayorga Usero
CFGS Desarrollo de Aplicaciones Web
I.E.S. Maestre de Calatrava - Ciudad Real
Desarrollo Web en Entorno Cliente
UT04 Práctica 1
*/

// Función de testeo.
function testList()
{
    // Primero creo unos cuantos objetos Book para probar los métodos de la Lista que crear
é después.
    let book1=new Book("978-84-9804-654-
0", "El Quijote", "Miguel de Cervantes", new Date(1605, 0, 1), 20);
    let book2=new Book("978-84-6703-338-
3", "Hamlet", "William Shakespeare", new Date(1609, 0, 1), 15.6);
    let book3=new Book("978-15-1965-334-
5", "Orlando furioso", "Ludovico Ariosto", new Date(1532, 0, 1), 21.02);
    let book4 = {
    title: "Tartufo",
    author: "Molière",
    publicationDate:
    new Date(1669, 0, 1),
```

```
    price: 9.17, };
    let book5=new Book("978-84-4142-514-
9", "Divina comedia", "Dante Alighieri", new Date(1472, 0, 1), 14.25);
    let book6=new Book("978-84-6703-458-
5", "El perro del hortelano", "Lope de Vega", new Date(1618, 0, 1), 6.60);
    let book7=new Book("978-95-0039-253-
2", "Leviathan", "Thomas Hobbes", new Date(1651, 0, 1), 6.60);
    // Ahora creo una lista de con un número máximo de ítems de 5.
    lista=new ListaOrdenadaPorFecha(5);
    // Compruebo si la lista está vacía con la función isEmpty.
    console.log("Compruebo si la lista está vacía con la función isEmpty.");
    console.log(lista.isEmpty());
    // Compruebo si la lista está llena con la función isFull.
    console.log("Compruebo si la lista está vacía con la función isFull.");
    console.log(lista.isFull());
    // Añado los tres primeros libros con la función add y los muestro por consola con la f
unción toString.
    console.log("Añado los tres primeros libros con la función add y los muestro por consol
a con la función toString.");
    lista.add(book1);
    lista.add(book2);
    lista.add(book3);
    console.log(lista.toString());
    // Voy a intentar añadir el book4 'Tartufo', que he creado de manera literal, y no tien
e ISBN. Debe saltar la excepción personalizada que he creado.
    console.log("Voy a intentar añadir el book4 'Tartufo', que he creado de manera literal,
 y no tiene ISBN. Debe saltar la excepción personalizada que he creado.");
    lista.add(book4);
    // Devuelvo el número de elementos en la lista con la función size.
    console.log("Devuelvo el número de elementos en la lista con la función size.");
    console.log(lista.size());
    // Añado el book5 'Divina comedia' en la posición 1 (recordar que el primer elemento de
 un array es el 0) con la función addAt. Se muestra el nuevo tamaño de la lista.
    console.log("Añado el book5 'Divina comedia' en la posición 1 (recordar que el primer e
lemento de un array es el 0) con la función addAt. Se muestra el nuevo tamaño de la lista."
);
  console.log(lista.addAt(book5, 1));
    console.log(lista.toString());
    // Obtengo el segundo elemento de la lista con el método get.
    console.log("Obtengo el segundo elemento de la lista con el método get.");
    console.log(lista.get(1));
    // Hallo el índice del objeto book3 'Orlando furioso', con la función indexOf.
    console.log("Hallo el índice del objeto book3 'Orlando furioso', con la función indexOf
.");
    console.log(lista.indexOf(book3));
    // Intento hallar el índice del objeto book6 'El perro del hortelano', que no está en l
a lista.
    console.log("Intento hallar el índice del objeto book6 'El perro del hortelano', que no
 está en la lista.");
    console.log(lista.indexOf(book6));
    // Hallo el índice del objeto book3 'Orlando furioso', comenzando a buscar por el final
, con la función lastIndexOf.
    console.log("Hallo el índice del objeto book3 'Orlando furioso', comenzando a buscar po
r el final, con la función lastIndexOf.");
    console.log(lista.lastIndexOf(book3));
    // Muestro el máximo número de elementos que podemos tener en la lista con la función c
apacity.
    console.log("Muestro el máximo número de elementos que podemos tener en la lista con la
 función capacity.");
    console.log(lista.capacity());
    // Obtengo el primer elemento de la lista con la función firstElement.
    console.log("Obtengo el primer elemento de la lista con la función firstElement.");
    console.log(lista.firstElement());
    // Obtengo el último elemento de la lista con la función firstElement.
    console.log("Obtengo el último elemento de la lista con la función firstElement.");
```

```
    console.log(lista.lastElement());
    // Elimino el elemento de la lista que está en la posición 2 (recordar que el primer el
emento de un array es el 0) con la función remove.
    console.log("Elimino el elemento de la lista que está en la posición 2 (recordar que el
 primer elemento de un array es el 0) con la función remove.");
    console.log(lista.remove(2));
    // Muestro cómo queda la lista tras esta operación con la función toString.
    console.log("Muestro cómo queda la lista tras esta operación con la función toString.")
;
    console.log(lista.toString());
    // Elimino el elemento book3 'Orlando furioso' con la función removeElement.
    console.log("Elimino el elemento book3 'Orlando furioso' con la función removeElement."
);
    console.log(lista.removeElement(book3));
    // Muestro cómo queda la lista tras esta operación con la función toString.
    console.log("Muestro cómo queda la lista tras esta operación con la función toString.")
;
    console.log(lista.toString());
    // Ahora sustituyo el elemento en la posición 0 (recordar que el primer elemento de un
array es el 0) por el objeto book6 'El perro del hortelano'.
    console.log("Ahora sustituyo el elemento en la posición 0 (recordar que el primer eleme
nto de un array es el 0) por el objeto book6 'El perro del hortelano'.");
    console.log(lista.set(book6, 0));
    // Muestro cómo queda la lista tras esta operación con la función toString.
    console.log("Muestro cómo queda la lista tras esta operación con la función toString.")
;
    console.log(lista.toString());
    // Vacío la lista con la función clear.
    console.log("Vacío la lista con la función clear.");
    lista.clear();
    // Muestro cómo queda la lista tras esta operación con la función toString.
    console.log("Muestro cómo queda la lista tras esta operación con la función toString.")
;
    console.log(lista.toString());
}
window.onload = testList();
```